The National Virtual Observatory Book
ASP Conference Series, Vol. 382, © 2008
M. J. Graham, M. J. Fitzpatrick, and T. A. McGlynn, eds.

Chapter 8: Web-based Tools—CARNIVORE

Matthew J. Graham

Introduction

Registries are an integral part of the VO infrastructure, yet the greatest exposure that most users will ever need to have to one is discovering resources through a registry portal. Some users, however, will have resources of their own that they need to register and will go to an existing registry to do so, but a small number will want to set up their own registry. They may have too many resources to register with an existing registry; they may want more control over their resource metadata than an existing registry will afford; or they may want to set up a specialized registry, e.g. a subject-specific one. CARNIVORE is designed to offer those who want their own registry the functionality they require in an off-the-shelf implementation. This chapter describes how to set up your own registry using CARNIVORE.

1. Overview

CARNIVORE is a fully featured registry supporting publishing, searching and harvesting and is fully compliant with the IVOA Registry Interface specification (Chapter 41). Since VO resource metadata is encoded in XML, we decided when implementing CARNIVORE that it should employ a complete end-to-end XML approach to handling the metadata: this means that it is never mapped into a relational model for storage in tables or bound to object–oriented code representations as with other registry implementations. Consequently, CARNIVORE utilizes some of the latest XML technologies to achieve this:

XForms is the W3C specification for the next generation of web forms. It employs an MVC (model-view-controller) architecture to separate presentation from content and logic and allows on-the-fly validation. Traditional HTML forms hold data entered by the user through a form interface as a set of key-value pairs. When the form is submitted to a service, these pairs are then normally appended to its HTTP URL. XForms uses XML as its internal data model so that data entered by a user is automatically put into an XML document that can then be manipulated and submitted.

XQuery is the W3C specification for an XML query language (see Chapter 57). It is a fully functional, strongly typed language that provides the analogous query capability for XML data that the SQL language provides for relational data. Queries against XML data that can be tens of lines long in SQL and Java (to query a remote relational model-based registry) can be expressed in just a few lines of XQuery. XQuery also allows results to be returned in user-defined XML structures.

eXist is an open source native XML database with efficient indexing. Individual XML documents can be stored without alteration and grouped into hierarchical collections.

Orbeon Forms is an open source web application framework that underpins CARNIVORE. It employs XML pipelines written in XPL (XML Pipeline Language) to deliver all the functionality.

Despite using all of these, CARNIVORE deploys as a standard web application in a web servlet container such as Tomcat.

2. Installation

We are going to assume here that we are installing CARNIVORE into the Tomcat version that is included on the companion CD. As a first step, we need to setup our environment in the usual manner (see the Software section of the introduction to this book).

```
> source $NVOSS_HOME/bin/setup.csh
```

However, if you are installing CARNIVORE into an existing Tomcat instance then you will need to make sure that the `CATALINA_HOME` environment variable is set to the base path of the Tomcat installation.

2.1. The CARNIVORE Web Application

A JAR file containing the application can be found on the companion CD at `$NVOSS_HOME/java/CARNIVORE/CARNIVORE.jar` or downloaded from `http://nvo.caltech.edu/CARNIVORE.jar`. To install CARNIVORE, we will first need to stop any Tomcat instances we have running with the `stoptomcat` command. Now we will create a directory for CARNIVORE under the `webapps` directory of our Tomcat installation:

```
> cd $CATALINA_HOME/webapps
> mkdir registry-name
```

where `registry-name` is the name that we want to give to our registry. The public URL for the registry will be `http://<hostname>:8080/registry-name` where `<hostname>` is the fully-resolved name of the machine we're running on. Now we need to unpack the JAR file:

```
> cd registry-name
> jar xf /path/to/CARNIVORE.jar
```

where `/path/to/CARNIVORE.jar` is the location of the JAR file for CARNIVORE. Finally we can just restart Tomcat with the `starttomcat` command and point a local browser at `http://localhost:8080/registry-name` and we should see the front page of CARNIVORE.

2.2. CGI Interface

CARNIVORE also provides a Perl module to support an OAI HTTP GET interface (see Chapter 41) for harvesting (a prerequisite for this is that the Perl SOAP::Lite module is installed). If Tomcat is configured for CGI support (by default this is disabled but in the Tomcat version we have included on the CD, it is enabled) then nothing needs to be done and the URL for the OAI HTTP GET interface will be:

```
http://<hostname>:8080/registry-name/cgi-bin/OAI-XML/CARNIVORE/OAI.pl
```

However, if Tomcat is being run with CGI disabled then we can install the Perl module in another valid cgi-bin directory, such as one attached to an Apache web server. In this case, the URL for the OAI HTTP GET interface will be:

```
http://<path to cgi-bin>/OAI-XML/CARNIVORE/OAI.pl
```

where `<path to cgi-bin>` is the URL of the cgi-bin. We will cover the configuration of the CGI interface in the next section.

3. Configuration

There are a number of things that we need to do to configure our new installation of CARNIVORE before it can be used: we need to create some metadata which describes the registry and who is running it and change some security settings from their default values. We could also put some resource records into the registry and customize the front page and banner but these are not necessary to have a valid VO registry.

3.1. Registration

First we need to register ourselves as a user with our new CARNIVORE installation so select “Publish” from the taskbar at the top of the page and click on the “Register new user” button. As part of this process, we will also be creating the resource records for our registry, our organization (the affiliation that is being ascribed to this registry – this might be a department or a project) and its default authority: the first step to this is to decide what our organization is called. We see that the *Affiliation* dropdown currently shows `None` (as there are not yet any Organisation resources in the registry) and so we need to put the name of our organization in the “Other” input box. We can now fill in the remaining boxes on the form and click “Submit”.

We are now presented with some forms to fill in the metadata about our registry, our organization and its authority: we can toggle between the metadata for the three by pressing the “Registry”, “Organisation” and “Authority” buttons respectively. However, we must be careful not to click on the “Publish” button until we have filled in all our details for them. The name we gave our organization on the previous form now shows up as the *Title* of the Organisation resource record and the *Managing organisation* of the Authority resource record. Each piece of metadata has an associated tooltip describing it in greater detail that can be brought up by dragging the mouse

over the question mark icon for that metadata. Input boxes which have a pink background indicate fields that are required: these must be filled in for the "Publish" button to submit the form details, filling them in will remove the pink background.

The first field that we need to fill in is the *Identifier* for the Organisation. This needs to conform to the IVOA specification for resource identifiers (see Chapter 41) and the red exclamation mark after the tooltip help icon indicates that the current value of the field does not: validation of the field value only occurs when another input box is selected. The syntax needs to be:

```
ivo://<authority ID>/<resource key>
```

where the *authority id* will be used as the *Identifier* value for the Authority and in the *Identifier* value for the Registry: we suggest a domain name style authority, e.g. nvo.caltech or sdss.org.

Next we should fill in the details for the registry interfaces under the "Capability data" tab on the Registry record. There are two URLs that we need to provide for the Harvest interface: the "HTTP GET Access URL" is the URL of CGI-based OAI interface that we installed in Section 2, and the "SOAP Access URL" is the URL of the OAI web service interface which will be:

```
http://<hostname>:8080/<registry name>/services
```

Finally we can fill in at least the rest of the mandatory fields for the Registry, the Organization and its Authority and then click on "Publish".

3.2. OAI Interfaces

We need to set up the identity of the registry that will be returned in response to an OAI *Identify* query. As mentioned in Section 2.2, the OAI HTTP GET interface also requires the Perl SOAP::Lite module as it is really just a proxy to the SOAP interface, and so we need to specify the endpoint of the SOAP interface to the OAI HTTP GET interface.

The data used in the response for the HTTP GET interface is in the `config.xml` file in the subdirectory `OAI-XML/CARNIVORE` under the cgi-bin: for Tomcat, this will be: `$CATALINA_HOME/webapps/registry-name/WEB-INF/cgi` but if we installed the OAI Perl module elsewhere then it will be wherever we installed it. The three elements that need to be changed are:

```
<repositoryName>Name of the registry</repositoryName>
<adminEmail>Email of registry administrator</adminEmail>
<archiveId>A short name identifying the registry</archiveId>
```

The location of the SOAP interface is also set by the value of the element:

```
<dburi>http://localhost:8080/registry-name/services</dburi>
```

For the SOAP interface, only the `<repositoryName>` and `<adminEmail>` elements need to be changed in the `$CATALINA_HOME/webapps/registry-name/WEB-INF/resources/ws/identity.xml` file.

Finally we need to change the URL for the OAI interface that is mentioned on the web page for the Harvest section (see Section 4.6). This can be found on line 38 of the file: `$CATALINA_HOME/webapps/registry-name/WEB-INF/resources/harvest/harvest.xhtml` and we set it to the value we used in Section 2.2.

3.3. Setting the Administration Passwords

There are two administration passwords that should be changed from their default values: that for the `admin` user in the eXist database (default is blank), and that for the CARNIVORE `administrator` user (default is administrator). To reset these, we first need to stop Tomcat with the `stoptomcat` command and then change to the `tools` subdirectory of the CARNIVORE installation:

```
> cd $CATALINA_HOME/webapps/<registry name>/WEB-INF/resources/tools
```

Then we need to start the eXist database client GUI:

```
> java –Dexist.home=. –jar start.jar
```

Under the “Type” dropdown, select Embedded and then enter the following for the “Configuration”:

```
<$CATALINA_HOME value>/webapps/<registry name>/WEB-INF/exist-conf.xml
```

where <$CATALINA_HOME value> needs to be replaced by the value of the $CATALINA_HOME environment variable.

First we’ll change the CARNIVORE administrator password, so double click on the “admin” resource and then “Users.xml”. This will bring up a window showing this document:

```
<Users>
   <user>
      <name>Administrator</name>
      <affiliation/>
      <email/>
      <uid>admin</uid>
      <password>administrator</password>
   </user>
   ...
</Users>
```

Change the value of the password element for the Administrator user and then click on the leftmost icon that resembles a floppy disk. Once we’ve done this, we can close this window.

Now we’ll change the eXist `admin` user password: under the `Tools` menu dropdown, select “Edit Users” to bring up a window listing the current users. Highlight the `admin` user and then add a password in the “Password” and “Password (re-

peat)" input boxes. Clicking on the "Modify User" button will change the password. We can then close the window and quit the application. Finally we should restart Tomcat with the `starttomcat` command.

3.4. Populating the Registry

CARNIVORE offers two ways of populating the registry: either by bulk loading a set of local resource record files (each of these must be an XML file conforming to the IVOA VOResource schema), or by harvesting from all the other current registries. To bulk load from a set of local files, we need to create an inventory file with the following format:

```
<files>
   <file>location of file</file>
   ...
</files>
```

We now select the "Admin" tab from the menubar at the top of a CARNIVORE page and specify the location of our inventory file in the "Location of inventory file" input box under the "Loading data" section. We also need to input the administration password (the password for the CARNIVORE Administrator user) at the top of the page and can then click on the "Bulk load data" button.

To harvest all the registries, we first need to make sure that CARNIVORE has an up-to-date list of existing registries. Again we select the "Admin" tab from the CARNIVORE menubar, enter the administration password and click on the "Get list of registries" button. This sends a query to the IVOA registry of registries for a current list of known registries. Once the list has returned, we select the "Harvest" tab on the CARNIVORE menubar, enter the administration password and can then either harvest a particular registry by clicking on the "Harvest" button next to it, or harvest from all the registries by clicking on the "Harvest all registries" button. Note that the first time this is done, it can take a while to harvest everything.

Automatic harvesting can be enabled by selecting the "Admin" tab, entering the administration password at the top of the page and then an interval between harvesting calls in milliseconds in the "Harvesting period (in milliseconds)" input box. The specified interval is the elapsed time between successive attempts to harvest all currently known registries. To start the scheduler, we just then click on the "Start harvesting" button and CARNIVORE will proceed to harvest from all currently known registries at the specified interval. This process will then run indefinitely or until the current Tomcat session terminates or the "Stop harvesting" button is pressed (again the administration password will be required). Note that an interval of 86400000 will harvest everything once a day.

3.5. Banner, front page and footer

It is quite likely that if we are setting up our own registry for a particular project or subject domain then we might want to personalize the banner, which appears at the top of each CARNIVORE page, and change the front page text of the default CARNIVORE installation. The banner is specified in the file

`$CATALINA_HOME/webapps/registry-name/WEB-INF/resources/banner.xml` and the front page in `$CATALINA_HOME/webapps/registry-name/WEB-INF/resources/index.xhtml`. Each page also carries a footer with contact details for feedback and copyright information. The footer is specified in the file `$CATALINA_HOME/webapps/registry-name/WEB-INF/resources/footer.xml`.

4. Using CARNIVORE

Within the banner at the top of every page in CARNIVORE, there is a taskbar and in this section, we will briefly explore each of the different subsections that are linked to on it. Nominally each of these represents a different piece of registry functionality.

4.1. Home

The default CARNIVORE home page gives a brief overview of the capabilities of the registry and a version history. Section 3.5 gives details on how it can be personalized.

4.2. Summary

The Summary page presents a census of the registry contents organized by resource type (see Chapter 41). This is similar to the Contents page of the STScI/JHU Registry. Note that the types listed here refer not only the values of `xsi:type` that may appear on the `<Resource>` element (e.g. Data Collection) but also on `<capability>` elements (e.g. Cone Search) in a resource record. This means that the same resource might appear under multiple groupings, e.g. a data collection with both a Cone Search and Simple Image Access interface would appear three times, once under each different typing.

Pressing the “Browse” button next to a resource type will take us to an expanded view of the resource records of that type in the registry. The title, first 50 words of the description and subject content for the first 20 relevant records are shown and the next and previous sets of 20 such records can be displayed with the navigation arrows. Next to each record listing there is a “Raw XML” button which will show you the full VOResource record for this resource.

Record listings for Cone Search and Simple Image Access resource types will also have a “Try It” button next to each entry and this will take us to a sample client form that can send a request to that particular service. Note that the form derives all its parameters from the resource record for a service and so any inaccuracies in the record will be reflected in the sample client. At the top of the form, the registry where the record originated (i.e. from which it was harvested) is given so that in such cases, we could contact the curator of the registry and notify them of an inaccurate record.

4.3. ResolveId

The ResolveId page is a simple form to allow the resolution of an IVOA identifier. Entering the identifier in the input box and then pressing “Resolve” will retrieve the specified record from the registry and display it.

4.4. Publish

The CARNIVORE publishing interface shares much in common with the NCSA registry publishing interface that is described in Chapter 44 (which should make the eventual move to a common publishing interface for all registries that much easier). Firstly to publish a resource, we need to be a registered user of the registry so the first page of the CARNIVORE Publish subsection is a login page. Normally we would just enter our username and password but if we need to register ourselves as a new user, see Section 3.1 for a description of the process. Unlike the NCSA registry, CARNIVORE has no explicit mechanism for trying out the publishing interface without actually publishing the data.

> If we want to use a particular Authority ID in the identifier for the resource we want to publish, we have to publish the resource in the registry where the resource record for the Authority was created: e.g. the Authority ID `nvo.caltech` was created at the Caltech registry so if someone wants to use this authority in their identifier then they would have to register their resource at Caltech.

Once we have logged in (and possibly created Organisation and Authority resource records if we are a new user), both registries present us with a list of the resources that we have already published in the registry (and in the NCSA case, also those that we have yet to commit). These resources can be edited (but we cannot change the resource identifier or type), copied (giving us a resource of the same type and with the same Authority ID in the identifier) and deleted.

To create a new resource record, we need to select one of the Authority IDs available to us – this will be based on which Organisation we affiliated ourselves with when we registered as a user – and the relevant Resource Type. If we choose to create a "Catalog Service" then we can also add a "Cone Search" and/or a "Simple Image Access" Capability. Note that this is how these resource types are expressed under the new VOResource v1.0 schema.

Whether we are editing, copying or creating a new resource, we will be presented with the appropriate form for our resource type (if we created an Organisation and Authority when we registered ourselves as a user then we will have already encountered these). Each piece of resource metadata has an associated tooltip describing it in greater detail that can be brought up by dragging the mouse over the question mark icon for that metadata. As with the NCSA interface, required information is indicated by color but CARNIVORE uses a pink background on the input and text boxes for required fields: these must be filled in for the "Publish" button to submit the form details and filling them in will remove the colored background. Similarly certain fields have formatting constraints, e.g. identifiers are not allowed to contain certain characters, and if the information entered does not comply with these then this will be indicated by a red exclamation mark next to the field. Again the "Publish" button will not submit the form details if there are any such errors with form. Note that the validation checks and help facility are a consequence of employing XForms technology for the forms.

If, at any stage, we are interested in seeing what the VOResource XML document that we are creating looks like then we just need to click on the "View XML"

button. If all that XML is too disconcerting then we can get rid of by clicking on the "Hide XML" button.

Finally when we are ready to publish our resource, we just click on the "Publish" button which will enter the VOResource XML document into the eXist database and take us back to the page listing our resources in the registry. We should now see our new resource amongst the list of resources. Any other registry that now harvests this registry will pick up the new resource and so it will propagate out into the VO.

4.5. Query

CARNIVORE has a number of querying interfaces: the default one presented to us on the Query page is a form for performing basic text searches – this is essentially a more advanced version of the basic keyword search interface of the STScI/JHU registry described in Chapter 42. We can select whether we want to restrict our search to certain resource types; which text fields, e.g. title or description, we want to search on and matching criteria, e.g. starts with or contains a phrase; and we can constrain our search to resources with specific content types or content levels. The page displaying the query results is exactly the same as encountered when browsing by resource type under the Summary subsection (see Section 4.1): the title, first 50 words of the description and subject content for the first 20 relevant records are shown and the next and previous sets of 20 such records can be displayed with the navigation arrows. The "Raw XML" and "Try It" buttons also feature where appropriate.

There is also an "Advanced Query" interface that allows arbitrary XQuery/XPath (see Chapter 57) and ADQL (see Chapter 36) queries to be submitted against the registry which can be found by following the "XQuery/XPath" and "ADQL" links as the top of the Query page. In contrasted to the "Advanced" interface of the STScI/JHU registry described in Chapter 42, which can take a SQL-like predicate based on resource metadata and just returns matching resources, XQuery allows us to select fragments of resources, e.g. just the Capability elements, and combine these into new XML documents which are the query results.

The XQuery page has a number of example queries that we can select (and edit), a set of drop boxes to help us build up a query and a textbox where we can type in our own query – note that both any example query or built query will appear in this textbox. We can also select previous queries that we may have submitted this session and choose how many results we want appearing on each query return page. The ADQL page is much simpler with just a textbox where arbitrary ADQL/X queries can be submitted.

4.6. Harvest

The Harvest page shows a log of which registries this registry is harvesting from and and how and when each was last harvested. It also allows us to harvest directly from a particular registry or all registries by entering the "Administration password" (see Section 3.3) and then clicking on the "Harvest" button next to the particular registry or the "Harvest all registries" button respectively. Note that this harvesting will occur in addition to any automatic scheduled harvesting (see Section 3.4).

4.7. Admin

The functionality of the Admin page is described in Section 3.4.

4.8. Web Services

Web service interfaces (see Chapter 59) are designed for programmatic clients and not humans. However, one of the useful features of the .NET framework that underpins virtually all Windows-based web services is that any exposed web service has a test form associated with it which a human can use to try it out and CARNIVORE has adopted this functionality under its Web Services page (note that CARNIVORE supports the new standard Registry Interface specification described in Chapter 43). Each of the web service operations listed links to a test form for that operation. The form has input boxes for each of the arguments of the operation and shows what the SOAP request and response messages look like. If the service is actually invoked by pressing the "Invoke" button then the response message will be updated with the result from the service.

It should be noted that the web service interface is implemented entirely in XML (XSLT and XPL) without any code-binding frameworks such as Apache Axis.

Useful Links

W3C XForms page. Nov 2006; http://www.w3.org/MarkUp/Forms/ [Accessed 12 Dec 2006]
eXist. Nov 2006; http://exist.sourceforge.net/ [Accessed 12 Dec 2006]
Orbeon Forms. Dec 2006; http://www.orbeon.com/ [Accessed 12 Dec 2006]